

OWASP Top 10 Security Challenge

Below is a vulnerable Python script. Read through it carefully and identify the 5 security vulnerabilities.

Mark the correct checkboxes.

```
import os
import sqlite3
import jwt
import requests
import ldap3

# Security Flaw 1: Broken Access Control (No authentication check)
def view_admin_data():
    conn = sqlite3.connect("database.db")
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM admin_data") # No access control
    results = cursor.fetchall()
    for row in results:
        print(row)
    conn.close()

# Security Flaw 2: LDAP Injection (User input directly in LDAP filter)
def ldap_authenticate():
    server = ldap3.Server("ldap://example.com")
    conn = ldap3.Connection(server)
    user_input = input("Enter username: ")
    ldap_filter = f"(uid={user_input})" # LDAP Injection vulnerability
    conn.search("dc=example,dc=com", ldap_filter)
    print(conn.entries)

# Security Flaw 3: Hardcoded JWT Secret (Cryptographic Failures)
JWT_SECRET = "supersecretkey" # Hardcoded secret key
def generate_token(username):
    return jwt.encode({"user": username}, JWT_SECRET, algorithm="HS256")

# Security Flaw 4: Insecure Deserialization
def decode_token(token):
    return jwt.decode(token, JWT_SECRET, algorithms=["HS256"], options={"verify_signature": False}) # No signature verification

# Security Flaw 5: SSRF (Server-Side Request Forgery)
def fetch_data():
    url = input("Enter API URL: ") # No URL validation
    response = requests.get(url) # Can be used to access internal services
    print(response.text)
```

Which 5 vulnerabilities are present? (Check the correct ones)

- ☐ A. Broken Access Control
- ☐ B. Cross-Site Scripting (XSS)
- ☐ C. LDAP Injection
- ☐ D. Cryptographic Failures (Hardcoded Secret Key)
- ☐ E. Insecure Design
- ☐ F. Security Misconfiguration
- ☐ G. Insecure Deserialization
- ☐ H. Server-Side Request Forgery (SSRF)
- ☐ I. Using Components with Known Vulnerabilities
- ☐ J. Cross-Site Request Forgery (CSRF)